AD-A074 937 AIR FORCE MANPOWER AND PERSONNEL CENTER RANDOLPH AFB TX F/G 12/1

HOW TO SPEED UP YOUR REGRESSION MODEL.(U)

MAY 79 T M BEATTY, K P JAMES

UNCLASSIFIED NL

1 OF 1

AD
A074937

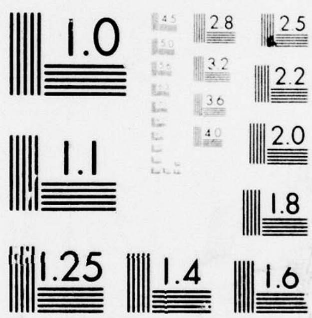
END

DATE

FILMED

11-79

DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) How to Speed Up Your Regression Model.		5. TYPE OF REPORT & PERIOD COVERED (9) Final rept.,
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) T.M. / BEATTY K.P. / JAMES		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Directorate of Personnel Data Systems Air Force Manpower and Personnel Center Randolph AFB, Texas 78148		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS HQ Air Force Manpower and Personnel Center Randolph AFB, Texas 78148		12. REPORT DATE (11) May 1979
		13. NUMBER OF PAGES ~~32~~ (12) 34
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

LEVEL II

(12)

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

regression	correlation	BMDO2R
multivariate regression	intercorrelation	BASIS
least squares	matrix	
ordinary least squares	SPSS	
	Biomedical Computer Programs	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Air Force personnel managers must be able to accurately forecast the force size. This need is explicit in meeting statutory budget limitations. Further, officer losses drive accessions, training, and promotion; thus the need for accuracy in forecasting losses cannot be over-emphasized. To accomplish this objective, loss rates have been generated using Ordinary Least Squares (OLS) stepwise regression run on what are locally dubbed the "binary files". The purpose of this paper is to report a front-end processor to OLS which has reduced computer run time by 85 percent for this organization.

DD FORM 1 JAN 73 1473

411 402 UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

HOW TO SPEED UP YOUR REGRESSION MODEL

TECHNICAL MEMORANDUM

BY

T.M. BEATTY

K.P. JAMES

MAY 1979

SYSTEMS SOFTWARE AND DEVELOPMENT BRANCH
SYSTEMS DEVELOPMENT AND SUPPORT DIVISION
DIRECTORATE OF PERSONNEL DATA SYSTEMS
AIR FORCE MANPOWER/PERSONNEL CENTER
RANDOLPH AFB, TEXAS 78148

HOW TO SPEED UP YOUR REGRESSION MODEL

TECHNICAL MEMORANDUM

BY

T.M. BEATTY
K.P. JAMES

MAY 1979

APPROVED BY:

FREDERICK S. PHILLIPS, SR., COL, USAF
DIRECTOR, PERSONNEL DATA SYSTEMS

While the contents of this report are considered to be correct, they are subject to modification upon further study. This report does not promulgate official Air Force policies or positions. The technical conclusions are solely those of the authors.

FOREWORD

This report and the associated software were prepared by the Modeling Section of the Air Force Manpower and Personnel Center in response to the need to solve large multivariate regression problems (100 attributes with up to 30,000 observations) in the management of the one million plus personnel employed by the Air Force. The collaboration of the following people is acknowledged: LCDR C. Pennington, Lt D. Hemphill, CMS L. Staton, and TSG D. Francis.

EXECUTIVE SUMMARY

Problem

Air Force Personnel managers must be able to accurately forecast the force size. Taken in a more general context, managers must be able to forecast a system response to independent stimuli . Ordinary Least Squares (O.L.S.) multivariate regression has been used to meet this need. O.L.S. Regression consumes inordinate ADP resources.

Objective

Reduce ADP resource usage in regression studies.

Approach

Isolate the portion(s) of regression which account for the greatest resource consumption and optimize.

Results

The major portion (75-95 percent) of the ADP resource was found to be expended in the computation of intercorrelation matrices. Since many regression problems are sparse due to the use of dummy variables, the introduction of logic to omit zeroe attribute values within an observation has provided ADP resource savings as high as 90 percent.

Conclusions

1. Regression problems of less than 91.5% density will yield economies.
2. Data files should not be processed with general utility regression programs.
3. As a second choice process data files with stand-alone correlation matrix builders.
4. The greatest economies will be realized if no data file is created for regression analysis. Rather insert correlation matrix build logic in the ADP system at the point where regression file(s) would be created.
5. Any of several matrix input regression packages may then be used to perform the actual stepwise or multiple regression.

ABSTRACT

Air Force personnel managers must be able to accurately forecast the force size. This need is explicit in meeting statutory budget limitations. Further, officer losses drive accessions, training, and promotion; thus the need for accuracy in forecasting losses cannot be over-emphasized. To accomplish this objective, loss rates have been generated using Ordinary Least Squares (OLS) stepwise regression, run on what are locally dubbed the "binary files". The purpose of this paper is to report a front-end processor to OLS which has reduced computer run time by 85 percent for this organization.

INTRODUCTION

Numerous software packages are available to solve the multivariate regression problem. To wit ; SPSS Incorporated's statistical package for the social sciences, the Biomedical Division's BMD02R, Burroughs' Advanced Statistical Inquiry System (BASIS), and Greenberger & Ward's Iterative Method. These methods and presumably a host of others capitalize upon the ability to sequentially consume virtually an unlimited amount of data, reduce the data to a correlation matrix, and "solve the problem". This feature of sequential processing of data into a relatively small core-resident matrix is one of the main selling points for OLS regression. However, the run time of most regression packages is notorious. In fact, the literature, both proprietary and public, note that providing a correlation matrix as input to the regression program will significantly reduce the regression run time. In application this approach saves time only if the user wishes to run various "sub-problems" against the same data file. (For example one may wish to make individual runs with the same independent variables against two or more dependent variables.) The lack of savings observed in running the single problem results because the data file must still be sequentially processed to build the correlation matrix. This leads to the incontrovertible conclusion that in order to make any money in reducing regression run time, the correlation matrix generation logic must be attacked. This, then is our approach.

This paper reports three methods which have resulted in significant computer resource savings in our application of O.L.S. It is presumed that readers of this paper are generally conversant with O.L.S. methodology. Therefore the objective of this paper will be to provide techniques which may provide reductions in run time, and not a tutorial on regression or correlation derivation.

METHOD

General

A straight-forward method of doing regression is to sequentially read the observations into an array, say VAR_i where i varies from 1 to the number of attributes and then execute code similar to the following:

```
SUM(I) = SUM(I) + VAR(I)
SSUM(I) = SSUM(I) + VAR(I) * VAR(I)
XYSUM (I,J) = XYSUM (I,J) + VAR (I) * VAR (J)
for I = 1 to number of attributes
    J = 1 to number of attributes
```

After processing all observations as above, then compute the means, standard deviations, and the intercorellation matrix as below: Where N = the number of observations.

$$XMEAN(I) = SUM(I)/N$$

$$STDEV(I) = \left(\frac{SSUM(I) - N*XMEAN(I)*XMEAN(I)}{N-1}\right) ** 0.5$$

$$R(I,J) = \frac{XYSUM(I,J) - SUM(I)*SUM(J)/N}{(N-1)*STDEV(I)*STDEV(J)}$$

The above produces the means and standard deviations for the attributes and the Pearson Product-Moment correlation coefficients which can be used as input to a properly designed regression program. But at what cost?

If the above logic were applied to a 100 attribute problem with 3 thousand observations, the SUM (I) and SSUM (I) computations would each be executed 300 thousand times taking approximately 15 seconds CPU time. The XYSUM (I,J) computation would be executed 30 million times at a cost of about 11 hundred seconds CPU time. All of this is in addition to the time required to read the file and handle various other statements required to complete the set of executable program logic. (all timing estimates are for a Burroughs B6700).

There are obvious savings in the XYSUM (I,J) computations since the XYSUM matrix is symetric to the diagonal. Automatically the cost can be reduced by 49.5% or to 556 seconds of CPU time. Additionally the diagonal of the correlation matrix contains only 1's; therefore that XYSUM on the diagonal is extraneous. This reduces the time required to 545 seconds CPU. These economies are recognized by most, but not all, statistical packages which provide O.L.S. regression. There are, however, potential problems with the approach above.

Computers are limited in the number of significant digits that can be represented in a real number. And in regression we frequently deal with big numbers. Specifically, on the AFMPC B6700 a real number contains 11 significant digits. Thus truncation errors may develop if the sums, sums of squares, or cross-product sums exceed 10 to the 11th power. Depending upon cirmcumstances, this could result in attempting to compute the square root of a negative number or simply erroneous standard deviations and correlation elements. Since many regression

algorithms have not planned for this anomaly, the researcher is cautioned to consider this possibility when dealing with big numbers, particularly if the attempted regression run terminates with "INVALID ALOG ARGUMENT".

Thus far we've reduced the computation of XYSUM by over one-half for those statistical packages which do not recognize the symmetry of the matrix. The next area and the most important is the database itself. The data files we utilize daily are files created from raw data, the Master Personnel File (MPF). The raw data is converted to what we call "binary files". Each observation contains upwards of 100 variables or attributes. Most, but not all, attributes are "1" or "0", e.g., either the individual has a regular commission or he does not. Our new composite binary file contains on/off state variables and continuous variables, such as age. No statistical packages yet observed recognize the economics of checking for a value of "0" for an attribute before doing the SUM (I), SSUM (I), md XYSUM (I,J) computations. Our point is; a "0" added to a value of SUM (I) is the original SUM (I) and that squared, is the original SSUM (I), so why do them? By inclusion of 2 lines of code (in most cases), we check to see if the value is "0", increment the counter by one, and step back to process the next value instead of going thru all the calculations to end up eith the same results. We use the same logic for XYSUM (I,J). The following is a sample of the code from our local regression package. The starred lines are the added code.

```
      DO  80  J=1, KK
      *IF(VAR(J).EQ.O) GO TO 80
      SUM (J) = SUM (J) + VAR(J)
      SSUM (J) = SSUM (J) + VAR (J) * VAR (J)

      DO 110 K=1, J-1
      *IF (VAR(K).EQ.0) GO TO 110
      XYSUM (J,K)=XYSUM (J,K) + VAR (J) * VAR (K)
  110 CONTINUE
  80 CONTINUE
```

The added lines of code will cost us some CPU time. The overall CPU runtime reduction is clearly a function of the file density A test problem was constructed to see at what point would the modifications to the regression break even with the regression run time. To accomplish this the inner and outer DO statements were timed in a variety of computer mixes on the B6700. To process a 10,000 observation problem with 100 variables; the average run time was:

outer computations =	56 seconds
inner computations =	2200 seconds
Total	2256 seconds

The worst case with the new regression code and an 80% dense file was:

outer computations	45 seconds
inner computations	2216 seconds
Total	2261 seconds

If a file is 80% or less dense then the inclusion of the extra lines of code is cost effective.

To be able to reduce CPU time enables drastic reductions in clock time. Clock time represents a tie up of computer resources and personnel. Two actual files with 28% density are presented below:

a. File size: 2374 observations
   Attributes: 92 variables
   File Density: 28%
   File Format: Formatted
   CPU run time: 1375 seconds versus 230 seconds
   Clock time: 1 hr, 9 min, 57 sec versus 6 min 58 sec

b. File size: 4197
   Attributes: 94
   File Density: 28%
   File Format: Formatted
   CPU run time: 2676 sec versus 322 sec.
   Clock time: 2 hrs, 15 min 55 sec versus 10 min 46 sec

In example a., the CPU run time was reduced 83% while the clock time was reduced 90%. In example b. the CPU run time was reduced 87% while the clock time was reduced 96%. This is significant savings as the density of our average files is 28%. To restate the objectives we had; we needed to run regressions with matrix input and needed to speed up the matrix build process. The savings in CPU and clock time is geometric. Once a parent binary file is created capturing data at given points in time numerous regressions can be run on the file utilizing subsets of variables to answer a variety of questions in the same or less amount of time it took to run only one regression.

Application

First and foremost, the user must avoid processing the raw data with off-the-shelf, generalized software. And, if reading the data file can be avoided all together, even better. Locally we have modified production regression applications in both of the manners indicated.

In the first case our regression program was modified at the point where the data file would be sequentially consumed. At this point a procedure is invoked which processes all of the raw data and returns the intercorrelation matrix, the means, and the standard deviations to the regression program. Using the procedure shown below in figure #1, we have realized savings of 75-85 percent in CPU and in excess of 90 percent in elapsed time processing data files which average 28 percent density. This specific application has been generalized to produce a utility procedure which may be invoked by any calling program requiring the means, standard deviations, and a correlation matrix. Appendix A presents the utility procedure in ALGOL. As can be readily seen from the program documentation, this procedure may be included in a calling program to process observations as sequentially read, or introduced earlier in a system. It is this latter application which provides the greatest efficiency.

BURROUGHS B6700 ALGOL COMPILER, VERSION 2.9.190,   FRIDAY, 02/02/79, 12:16 PM.

O B J E C T / L O S S 1 / L D A T A

FIG 1

```
$SET LINEINFO LIST
%
% PROCEDURE TO READ DATA FOR REGRFUN
%
%
%    INSERT INTO MAIN PROG :  $SET AUTOBIND
%                             IWIND=0
%                             IF (WIND.EQ.YES) IWIND=1
%                             CALL LOADATA(IWIND,ISKIP,IREC,IPROB,
%                                    N,IP,X,STDEV,XMEAN,A)
%
%
%
[FILE FILE3,FILE6,]
%
PROCEDURE LDATA(IWIND,ISKIP,IREC,IPROB,N,IP,X,STDEV,XMEAN,AA);
     VALUE IWIND,ISKIP,N,IP;
     INTEGER IWIND,ISKIP,IREC,IPROB,N,IP;
     ARRAY X[*], XMEAN[*], AA[*], STDEV[*];
BEGIN
                                                    LDATA IS SEGMENT 00002
DEFINE BINSIZE=132#;
FILE FILE3A(KIND=DISKPACK,PACKNAME="PRODATA.",FILETYPE=7,
            TITLE="BIN/LOSS75.");
                                                    DATA IS 0009 LONG
FILE FILE3B(KIND=DISKPACK,PACKNAME="PRODATA.",FILETYPE=7,
            TITLE="BIN/LOSS76.");
                                                    DATA IS 0009 LONG
FILE LINE(KIND=PRINTER);
                                                    DATA IS 0005 LONG
INTEGER I,I1,J,K,L,M,C2,C3,XI,CNT;
REAL T,T1,T2;
EBCDIC ARRAY BRECD[0:BINSIZE-1];
ARRAY TAG[0:0],SUM,SSUM[0:100];
LABEL PAR,EOF,EXIT;
DEFINE LDX(COL,LEN)=X[XI:=*+1]:=INTEGER(BRECD[COL],LEN)#,
        A[I,J]=AA[I+99*J]#,
        YRGRP=[38:7]#,
        FILNR=[31:2]#,
        RECNR=[29:30]#;
T2 := TIME(12);
IF NOT FILE3.OPEN THEN FILE3.OPEN:=TRUE;
%
%
%
IF IWIND GEQ 1 THEN BEGIN
     REWIND(FILE3);  WRITE(FILE6,</"INPUT FILE REWOUND."//>);  END;    2
%                                                                      2
%
IF ISKIP GTR 0 THEN BEGIN
     SPACE(FILE3,ISKIP);  IREC:=*+ISKIP;  IPROB:=*+ISKIP;              2
     WRITE(FILE6,<//I6," RECORDS SKIPPED."//>,ISKIP);  END;
```

```
%
%
J:=N-1;      M:=IP-1;
FOR I:=0 STEP 1 UNTIL J DO BEGIN
    READ(FILE3,1,TAG) [EOF:PAR];
    IF TAG[0].FILNR = 0
       THEN READ(FILE3A[TAG[0].RECNR],BINSIZE,BRECD)[EOF:PAR]
       ELSE READ(FILE3B[TAG[0].RECNR],BINSIZE,BRECD)[EOF:PAR];
    XI:=-1;
    FOR K:= 16 STEP 1 UNTIL  24,
            31 STEP 1 UNTIL  39,129,
            45 STEP 1 UNTIL  57,130,63,64,
            77 STEP 1 UNTIL  80,
            127
            DO LDX(K,1);
    LDX(65,2);  LDX(82,2);
    IREC:=*+1;   IPROB:=*+1;
    CNT := * + 1;
```

Accumulate data after each observation read.

```
    FOR K:=0 STEP 1 UNTIL M DO
        IF T:=X[K] NEQ 0 THEN
           BEGIN
              SUM[K] := * + T;
              SSUM[K] := * + T*T;
              FOR L:=0 STEP 1 UNTIL K-1 DO
                 IF X[L] NEQ 0 THEN
                    A[K,L] := * + T * X[L];
           END;
    END READ LOOP;
  C2 := CNT -1;
```

Compute mean, standard deviation, and correlation matrix after all data read.

```
  FOR K := 0 STEP 1 UNTIL M DO
    BEGIN
     T := XMEAN[K] := SUM[K]/CNT;
      T1 :=  STDEV[K] := ((SSUM[K]-CNT*T*T)/C2)**0.5;
      A[K,K] := 1;
      IF T1 NEQ 0 THEN
         FOR L := 0 STEP 1 UNTIL K-1 DO
            IF T := T1*STDEV[L] NEQ 0 THEN
               A[K,L] := A[L,K] := ((A[K,L]-SUM[K]*SUM[L]/CNT)/C2)/T
            ELSE  A[K,L] := A[L,K] := .0019
      ELSE FOR L := 0 STEP 1 UNTIL K-1 DO
            A[K,L] := A[L,K] := .0019;
    END;
T2 := TIME(12)-T2;
WRITE(LINE,*//,T2,CNT,M,K,L);
GO EXIT;
%
%
%
PAR:  WRITE(FILE6,<"PARITY ERROR ON INPUT AT RECD NR ",I8,
                  " THIS PROBLEM ",I6>,IREC,IPROB);
                                                          DATA IS 0012 LONG
      MYSELF.STATUS:=-1;
%
%
EOF:  WRITE(FILE6,<"UNEXPECTED END OF INPUT AT RECD NR ",I8,
                   " THIS PROBLEM ",I6>,IREC,IPROB);
                                                          DATA IS 000F LONG
      MYSELF.STATUS:=-1;
%
%
```

```
        EXIT:   END OF LOADATA;                                                       00008400  002:00F2:2
                                                                                          DATA IS 000F LONG
                                                                                          DATA IS 000A LONG
                                                                                     LDATA(002) IS 010B LONG
**** WARNING : THIS PROCEDURE VALID FOR BINDING ONLY  -  FOR OTHER USE RECOMPILE AT LEVEL 2  ****
                                                                                      BLOCK IS SEGMENT 000A
                                                                                     BLOCK(00A) IS 000C LONG
===========================================================================================================
NUMBER OF ERRORS DETECTED = 0.
NUMBER OF SEGMENTS = 9.  TOTAL SEGMENT SIZE = 360 WORDS.  CORE ESTIMATE = 1438 WORDS.  STACK ESTIMATE = 45
PROGRAM SIZE = 107 CARDS, 782 SYNTACTIC ITEMS, 37 DISK SEGMENTS.
PROGRAM FILE NAME:  OBJECT/LOSS1/LDATA.
COMPILATION TIME = 13.966 SECONDS ELAPSED;  2.727 SECONDS PROCESSING;  1.203 SECONDS I/O.
===========================================================================================================
```

Our second application of these enhancements to regression has been accomplished by early introduction into an existing system of software similar to that above, in figure #1. In this case a system in which two files were merged produced a "binary" file for subsequent processing in a stand-alone regression package. In the system as it previously operated, the two files were merged and written to diskpack for later regression analysis. At this point program logic was added to the "merge" program to compute the necessary data for producing the means, standard deviations, and correlation matrix. This, thus saves the ADP time to write and read a file plus the storage medium to store data between processing steps. Resource usage was markedly reduced.

The implementation in the second case realized the preatest reduction in computer resources. Regression problems which had run in 1400 CPU seconds are now processed in 120 CPU seconds. In addition DASD storage can be reduced.

Appendix B presents the HOL source code illustrating one way to insert the time consuming portion of regression into a extant ADP application.

CONCLUSION/RECOMMENDATION

- OLS regression requires high CPU resources.
- Applied regression typically produces low density files. That is, introduction of dummy variables to accomodate non-linearity results in lots of zeros.
- Files of less than 80% density can be processed more rapidly by improved logic.
- Don't use off-the-shelf generalized regression packages to build correlation matrices.
- As a second choice use stand-alone correlation matrix generator.
- As a first choice hardwire correlation matrix logic into existing systems at the point where the data files for regression analyses are now produced.

Appendix A

WORKFILE: HEMPHILL/SRC/BOUND/MATRIX (04/27/79)　　　　4:41 PM FRIDAY, APRIL 27, 1979

```
$ SET ERRLIST LINEINFO
% W A R N I N G !   T H I S   P R O C E D U R E   M U S T   B E   B O U N D !
% ==================================================================
%  THIS PROCEDURE PRODUCES THE MEAN, STANDARD DEVIATION AND PEARSON'S
% PRODUCT-MOMENT INTERCORRELATION MATRIX FOR ANY NUMBER OF VARIABLES,
% AND ANY NUMBER OF OBSERVATIONS. THE MEAN, STANDARD DEVIATION AND
% INTERCORRELATION MATRIX ARE STORED IN A DISKPACK FILE WITH A TITLE
% COMPOSED OF A DIRECTORY OF "MATRIX/DATA/" AND A FILENAME SPECIFIED
% BY THE USER OR IN IT'S ABSENCE A FILENAME REFLECTING THE MONTH, DAY,
% YEAR, HOUR, MINUTE, AND SECOND THE FILE WAS CREATED
% (I.E. 01JAN99H00M01S01). ADDITIONALLY, THE PROCEDURE IS ABLE TO
% MINIMIZE THE IMPACT ON THE DEPENDENT VARIABLE OF TWO OR MORE HIGHLY
% CORRELATED VARIABLES. THE HIGH CORRELATION THRESHOLD CAN BE SET
% ANYWHERE FROM 0.0000 TO 1.0000 BASED ON A PARAMETER PASSED TO THE
% PROCEDURE AT EXECUTION.
%  THE USER'S PROGRAM SHOULD CONTAIN A DECLARATION LIKE THIS:
% PROCEDURE MATRIX(A,B); VALUE B; REAL B; ARRAY A[0]; EXTERNAL;
% FOR EACH OBSERVATION (I.E. RECORD READ), MATRIX SHOULD BE CALLED
% PASSING THE VALUES OF THE VARIABLES IN THE A ARRAY AND THE NUMBER OF
% VARIABLES (I.E. THE SIZE OF THE ARRAY) ONE-RELATIVE IN B. AFTER THE
% LAST OBSERVATION, MATRIX SHOULD BE CALLED ONE MORE TIME WITH THE HIGH
% CORRELATION THRESHOLD IN THE FIRST WORD OF THE A ARRAY AND THE
% FILENAME BEGINNING IN THE SECOND WORD OF THE A ARRAY. B SHOULD
% CONTAIN THE NUMBER OF THE DEPENDENT VARIABLE ZERO-RELATIVE.
%  ONE RESTRICTION IS THAT THE NUMBER OF VARIABLES WILL REMAIN THE
% SAME FOR EVERY OBSERVATION IN EACH GROUP OR SET OF OBSERVATIONS
% (I.E. EACH RECORD IN A FILE CONTAINS THE SAME NUMBER OF VARIABLES)
% THE FIRST TIME AND EACH SUBSEQUENT TIME THRU THE PROCEDURE WHILE B
% REMAINS THE SAME THE USER IS ACCUMULATING DATA. WHEN B CHANGES IT
% INDICATES THAT THE USER WANTS TO FINISH THE CALCULATIONS AND
% PRODUCE THE FILE CONTAINING THE MEAN, STANDARD DEVIATION AND THE
% INTERCORRELATION MATRIX.
%  THE FILE IS CREATED AS A FRONT END TO SPSS AND THEREFORE IS IN A
% FORMAT WHICH IT IS EXPECTING (I.E. EACH RECORD IS A CARD IMAGE. THE
% FIRST SET OF RECORDS CONTAINS THE MEANS OF THE VARIABLES, THE SECOND
% SET OF RECORDS CONTAINS THE STANDARD DEVIATIONS OF THE VARIABLES AND
% THE REST OF THE RECORDS CONTAIN THE INTERCORRELATION MATRIX. THE DATA
% IS WRITTEN WITH AN F10.5 FORMAT EIGHT WORDS PER RECORD. THUS IF THERE
% ARE MORE THAN EIGHT VARIABLES THE MEANS AND STANDARD DEVIATIONS WILL
% BE WRITTEN ON MORE THAN ONE RECORD, AND IF THERE ARE MORE THAN TWO
% VARIABLES THE INTERCORRELATION MATRIX WILL ALSO BE WRITTEN ON MORE
% THAN ONE RECORD. THIS IS USER INFORMATION ONLY.)
%  IN ORDER TO BIND THIS ROUTINE YOU MUST FIRST CREATE A COPY OF THE
% OBJECT CODE UTILITY/OBJ/MATRIX UNDER YOUR DIRECTORY. ADDITIONALLY YOU
% MUST SET AUTOBIND IN YOUR SOURCE CODE. FOR MORE COMPLETE INSTRUCTIONS
% READ EITHER GREGORY VOLUME II OR BURROUGHS BINDER MANUAL.
%  THIS IS A PRODUCT OF THE MODELING AND RESEARCH SECTION OF THE SYSTEM
% SOFTWARE AND DEVELOPMENT BRANCH AT THE AIR FORCE MANPOWER AND
% PERSONNEL CENTER (AFMPC/MPCDOP7). THE ORIGINAL CONCEPT WAS DEVELOPED
% BY MISTER THOMAS BEATTY, CHIEF MODELING AND RESEARCH SECTION. THE
% PROCEDURE WAS DESIGNED AND WRITTEN BY FIRST LIEUTENANT DAVID HEMPHILL,
% USAF. THE DESIGN AND PROGRAMING EFFORT WERE ACCOMPLISHED OVER A PERIOD
% FROM NOVEMBER 1978 THRU APRIL 1979. ANY INQUIRIES MAY BE DIRECTED TO
% EITHER MR BEATTY OR LT HEMPHILL AT AFMPC/MPCDOP7, RANDOLPH AFB TEX
% 78148 OR AT (COMM (512) 652- OR AUTO 487- & EXT 2233 OR 5684).
PROCEDURE MATRIX(VARIABLESARRAY,NUMBERVARIABLES);
  VALUE NUMBERVARIABLES; REAL NUMBERVARIABLES;
```

```
    ARRAY VARIABLESARRAY[0];
  BEGIN
    OWN ARRAY XYSUM[0:0],SUM,SSUM[0:0];
    OWN BOOLEAN NOTFIRST;
    OWN REAL ARRAYSIZE,OBSERVATIONS;
    REAL I1,I2,T1,T2,T3;
    IF NOTFIRST THEN                          %I N I T I A L I Z A T I O N II
    ELSE                                      %THIS PIECE OF CODE IS EXECUTED
    BEGIN                                     %THE FIRST TIME AND ONLY THE
      NOTFIRST:=TRUE;                         %FIRST TIME THROUGH FOR ANY ONE
      ARRAYSIZE:=NUMBERVARIABLES;             %SET OF DATA. UPON COMPLETING
      RESIZE(XYSUM[*],ARRAYSIZE*ARRAYSIZE);%WITH ONE SET OF DATA ALL THE
      RESIZE(SUM[*],ARRAYSIZE);               %OWN VARIABLES ARE DEALLOCATED
      RESIZE(SSUM[*],ARRAYSIZE);              %THUS ALLOWING THIS PROCEDURE
      OBSERVATIONS:=0;                        %TO BE CALLED AGAIN FOR A NEW
    END;                                      %SET OF DATA.
    IF ARRAYSIZE=NUMBERVARIABLES THEN                 %   M A I N  L I N E
    BEGIN                                             %THIS PIECE OF CODE IS
      OBSERVATIONS:=*+1;                              %EXECUTED EVERY TIME
      FOR I1:=0 STEP 1 UNTIL ARRAYSIZE-1 DO           %THE PROCEDURE IS CALLED
        IF T1:=VARIABLESARRAY[I1] NEQ 0 THEN          %FOR A PARTICULAR SET OF
        BEGIN                                         %DATA EXCEPT THE LAST
          SUM[I1]:=*+T1;                              %TIME. EXECUTING THIS
          SSUM[I1]:=*+T1*T1;                          %SMALL PIECE OF CODE
          FOR I2:=0 STEP 1 UNTIL I1-1 DO              %FOR EVERY OBSERVATION
            IF T2:=VARIABLESARRAY[I2] NEQ 0 THEN %RATHER THAN ALL THE
              XYSUM[I1*ARRAYSIZE+I2]:=*+T1*T2;        %CALCULATIONS WHICH ARE
        END;                                          %NOW DONE AS A FINAL STEP
    END                                               %SAVES CONSIDERABLE TIME.
                                                      %ADDITIONAL ECONOMIES ARE
                                                      %REALIZED IN CHECKING FOR
                                                      %ZERO VALUES IN VARIABLES
                                                      %WHEN SPARSE ARRAYS ARE
                                                      %INVOLVED.
    ELSE
    BEGIN
  % THE PROCEDURE IS CALLED ONE MORE TIME AFTER THE LAST OBSERVATION AND
  % THIS PIECE OF CODE IS EXECUTED TO PRODUCE THE FILE CONTAINING THE
  % MEAN, STANDARD DEVIATION, AND PEARSONS INTERCORRELATION MATRIX. IN
  % ADDITION HIGH CORRELATION VARIABLES ARE IDENTIFIED AND ELIMINATED
  % FROM THE EQUATION BASED ON A USER SUPPLIED TOLERANCE. THE USER MAY
  % SUPPLY A FILE NAME VIA THE REAL ARRAY BEGINNING IN THE SECOND WORD.
  % THE TOLERANCE IS IN THE FIRST WORD AND THE DEPENDENT VARIABLE ZERO
  % RELATIVE IS SUPPLIED IN THE SECOND PARAMETER.
      FILE OUTFILE(KIND=PACK,MAXRECSIZE=14,BLOCKSIZE=420,AREASIZE=120);
      FILE LINEOUT(KIND=PRINTER);
      ARRAY MEAN,STANDARDDEVIATION,HI1,HI2[0:ARRAYSIZE-1],WORK[0:15];
      REAL I3,I4,I5,T4;
      POINTER P;
      BOOLEAN HIT;
      EBCDIC VALUE ARRAY MON("JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC");
      TRUTHSET VLDCHRCTRS("0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ/");
      DEFINE
        DEPENDENT=NUMBERVARIABLES#,
        TOLERANCE=VARIABLESARRAY[0]#;
      SCAN P:VARIABLESARRAY[1] FOR T2:95 UNTIL IN VLDCHRCTRS;
      IF T2 NEQ 0 THEN
        REPLACE WORK BY "MATRIX/DATA/",P FOR T2 WHILE IN VLDCHRCTRS,"."
      ELSE
      BEGIN
```

```
      T1:=TIME(7);
      REPLACE WORK BY "MATRIX/DATA/",T1.[29:6] FOR 2 DIGITS,
        MON[(INTEGER(T1.[35:6])-1)*3] FOR 3,T1.[47:12] FOR 2 DIGITS,
        "H",T1.[23:6] FOR 2 DIGITS,"M",T1.[17:6] FOR 2 DIGITS,"S",
        T1.[11:6] FOR 2 DIGITS,".";
    END;
    REPLACE OUTFILE.TITLE BY WORK;
    T3:=OBSERVATIONS-1;
    FOR I1:=0 STEP 1 UNTIL ARRAYSIZE-1 DO
    BEGIN
      T1:=MEAN[I1]:=SUM[I1]/OBSERVATIONS;
      T2:=STANDARDDEVIATION[I1]:=((SSUM[I1]-OBSERVATIONS*T1*T1)/T3)
                                  **0.5;
      WRITE(LINEOUT,<I15,2E10.5>,I1,T1,T2);
      XYSUM[I1*ARRAYSIZE+I1]:=1;
      IF T2 NEQ 0 THEN
        FOR I2:=0 STEP 1 UNTIL I1-1 DO
          IF T1:=T2*STANDARDDEVIATION[I2] NEQ 0 THEN
          BEGIN
            T4:=XYSUM[I1*ARRAYSIZE+I2]:=XYSUM[I2*ARRAYSIZE+I1]:=
              ((XYSUM[I1*ARRAYSIZE+I2]-SUM[I1]*SUM[I2]/OBSERVATIONS)
              /T3)/T1;
            IF ABS(T4) >= TOLERANCE AND I1 NEQ DEPENDENT AND I2 NEQ
              DEPENDENT THEN
            BEGIN
              FOR I3:=0 STEP 1 UNTIL I5-1 DO
                IF HI[I3] = I1 THEN
                  HIT:=TRUE;
              IF NOT HIT THEN
              BEGIN
                HI[I5]:=I1;
                I5:=*+1;
              END;
              HIT:=FALSE;
              FOR I3:=0 STEP 1 UNTIL I5-1 DO
                IF HI[I3] = I2 THEN
                  HIT:=TRUE;
              IF NOT HIT THEN
              BEGIN
                HI[I5]:=I2;
                I5:=*+1;
              END;
              HIT:=FALSE;
            END;
          END
          ELSE
            XYSUM[I1*ARRAYSIZE+I2]:=XYSUM[I2*ARRAYSIZE+I1]:=0.0019
      ELSE
        FOR I2:=0 STEP 1 UNTIL I1-1 DO
          XYSUM[I1*ARRAYSIZE+I2]:=XYSUM[I2*ARRAYSIZE+I1]:=0.0019;
    END;
    FOR I1:=0 STEP 1 UNTIL I5-1 DO
    BEGIN
      HI2[I1]:=XYSUM[DEPENDENT*ARRAYSIZE+HI[I1]];
    END;
    FOR I1:=0 STEP 1 UNTIL I5-1 DO
    BEGIN
      T1:=ABS(HI2[I1]);
      FOR I2:=0 STEP 1 UNTIL I5-1 DO
        IF T2:=ABS(HI2[I2]) > T1 THEN
```

```
          BEGIN
            T1:=T2;
            I3:=I2;
          END;
        HI2[I3]:=0;
        FOR I2:=0 STEP 1 UNTIL I3-1,I3+1 STEP 1 UNTIL I5-1 DO
          IF ABS(XYSUM[HI1[I3]*ARRAYSIZE+HI1[I2]]) >= TOLERANCE THEN
          BEGIN
            WRITE(LINEOUT,<I6," REMOVED DUE TO HIGH CORRELATION WITH ",
              I4>,HI1[I2],HI1[I3]);
            FOR I4:=0 STEP 1 UNTIL HI1[I2]-1,HI1[I2]+1 STEP 1 UNTIL
              ARRAYSIZE-1 DO
              XYSUM[I4*ARRAYSIZE+HI1[I2]]:=XYSUM[HI1[I2]*ARRAYSIZE+I4]:=
                0.0017;
          END;
      END;
      WRITE(LINEOUT,*//,OBSERVATIONS);
      WRITE(OUTFILE,<8F10.5>,MEAN);
      WRITE(OUTFILE,<8F10.5>,STANDARDDEVIATION);
      FOR I1:=0 STEP 1 UNTIL ARRAYSIZE-1 DO
        WRITE(OUTFILE,<8F10.5>,FOR I2:=0 STEP 1 UNTIL ARRAYSIZE-1 DO
          XYSUM[I1*ARRAYSIZE+I2]);
      LOCK(OUTFILE);
      DEALLOCATE(XYSUM);
      DEALLOCATE(SSUM);
      DEALLOCATE(SUM);
      NOTFIRST:=FALSE;
    END;
  END;
```

HONEYWELL PAGE PRINTING SYSTEM- P1188-02

Appendix B


```
$ RESET LIST
$ SET ERRLIST LINEINFO
%*****************TSGT FRANCIS DPMODA EXT 2233
%**********CARDIN FORMAT
%     CC 1-5  BOARD NAME
%     CC 1-2     CY
%     CC   3     TYPE BOARD
%                   R=REG APPT
%                   N=NOM
%                   P=PERM
%                   T=TEMP
%     CC 4-5     GRADE BOARD OR TENURE OF APPT
%     CC 70-72   TERT ZONE CUTOFF
%     CC 74-76   SEC ZONE CUTOFF
%     CC 78-80   PRI ZONE CUTOFF
BEGIN
FILE BINARY (KIND=DISKPACK,PACKNAME="MPCDATA.",TITLE="BIN.",MYUSE=10,
             UNITS=CHARACTERS,FILETYPE=7);
FILE BOARDFILE (KIND=TAPE,BLOCKSIZE=6400,MAXRECSIZE=640,TITLE="BF.",
               UNITS=CHARACTERS,FILETYPE=7);
FILE LINE (KIND=PRINTER);
FILE CRD (KIND=READER,TITLE="CARDIN.");
FILE PRI(KIND=DISKPACK,PACKNAME="MPCDATA.",MAXRECSIZE=14,
        TITLE="TMB/MAT/TO6P.",BLOCKSIZE=2100,PROTECTION=SAVE,AREAS=10,
        AREASIZE=14),
     SEC(KIND=DISKPACK,PACKNAME="MPCDATA.",MAXRECSIZE=14,
        TITLE="TMB/MAT/TO6S.",BLOCKSIZE=2100,PROTECTION=SAVE,AREAS=10,
        AREASIZE=14),
     TER(KIND=DISKPACK,PACKNAME="MPCDATA.",MAXRECSIZE=14,
        TITLE="TMB/MAT/TO6T.",BLOCKSIZE=2100,PROTECTION=SAVE,AREAS=10,
        AREASIZE=14);
DIRECT ARRAY SUMS,SUMSS,SUMP,SUMPS,SUMT,SUMTT[0:100];
DIRECT ARRAY XYSUMS,XYSUMP,XYSUMT[0:10000];
ARRAY RECHOLD,MEANT,MEANS,MEANP,STDVT,STDVS,STDVP,IAPRI[0:100];
ARRAY IASEC,IATER[0:100];
EBCDIC ARRAY BIN[0:161],
             BF[0:639];
ARRAY DIST[0:3,0:100,0:2],
      CUTSCORE[0:3],                              %CUT OFF SCORE
      ZNCNT[0:3],
      ZONECNT[0:3],
      STATDIST1[0:3,0:200,0:1],            %ZONE,SCORE;SEL,NON-SEL;
      STATDIST2[0:3,0:1,0:2];               %ZONE;SEL,NON-SEL;>=<
BOOLEAN BOO; INTEGER PARCNT; LABEL PARR;
EBCDIC VALUE ARRAY TITL  ("       ","      ",
                          "PRIMAR","Y      ",
                          "SECOND","ARY    ",
                          "TERTIA","RY     ");
FORMAT CARDIN(X2,A1,A3,X63,A3,X1,A3,X1,A3);
FORMAT HDR1N("PREDICTED",X12,"PRIMARY ZONE");
FORMAT HDR1T("PREDICTED",X12,"PRIMARY ZONE",X13,"SECONDARY ZONE");
FORMAT HDR1R("PREDICTED",X12,"PRIMARY ZONE",X12,"SECONDARY ZONE",
             X11,"TERTIARY ZONE");
FORMAT HDR2N("  SCORE",X12,"TOTAL   SELECTED");
FORMAT HDR2T("  SCORE",X12,2("TOTAL   SELECTED",X10));
FORMAT HDR2R("  SCORE",X12,3("TOTAL   SELECTED",X10));
FORMAT HDR3(X14,"SELECT   > CUT   = CUT   < CUT      NON-SELECT   > CUT",
```

```
       "   = CUT   < CUT   PREDICTED CORRECTLY      PREDICTION RATIO");
FORMAT HDR4(A6,A6,"ZONE    PANEL",X11,"MEAN",X8,"STN DEV",X9,"SAMPLE");
FORMAT DET1(X24,3(I4,X4),X15,3(I4,X4),X8,I4,X19,F5.3);
FORMAT DET4(X23,I2,X10,F5.1,X10,F5.1,X10,I5);
ALPHA BOARD,OBBD;
REAL TVAP,TVAS,TVAT,VAP,VAS,VAT,VARAMT;
REAL TEMP,TEMS,TEMT;
INTEGER BDCNT,
        BDSCR,
        B,                          %BINARY MAXRECSIZE
        BDSCRSQ,                    %BOARD SCORE SQUARED
        BINCNT,
        BINCNT1,
        CUT,
        HOLD1,
        HOLD2,
        I,M,K,J,L,                    %NEW
        NRVARS,XX,XY,M1,M2,M3,            %NEW
        ISW,MRPRI,MRSEC,MRTER,            %NEW
        LSPRI,LSSEC,LSTER,
        IX1,
        PANEL,
        SAMPL,
        SCORE,
        SUMM,
        VAR,
        ZONE,
        X;
DEFINE ZONEDEF = INTEGER(BIN[141],1)#,      %BINZONE
       BINSEL = BIN[142]#,                  %BINSELECT
       BINPAN = BIN[143]#;                  %BINPANEL
LABEL READBIN,READBD,ENDRD,COMPARE,PASS1,ENDSTAT,
      ENDLOOP,READAGAIN,ENDAGAIN,ENDLOOP2,CKSCORE;
 $ PAGE
PROCEDURE HICOR(WCHARRY,WCHIA,WCHLS);
ARRAY WCHARRY[*],WCHIA[*];
INTEGER WCHLS;
  BEGIN
     LABEL OUTHI,ENDM;
     ARRAY R[0:100];
     REAL V;
     INTEGER MC,VALIA,LL,L,M,J,N;
     MC := 0;
     LL := (87-1)*NRVARS;                                %DEP VARIABLE
     FOR I := 1 STEP 1 UNTIL NRVARS DO
         R[I] := WCHARRY[LL+I];
     FOR M1 := 1 STEP 1 UNTIL WCHLS DO
       BEGIN
         IF MC = WCHLS THEN GO TO OUTHI;
         V := 0;
         FOR I := 1 STEP 1 UNTIL WCHLS DO
           BEGIN
             VALIA := WCHIA[I];
             IF ABS(WCHARRY[LL+VALIA]) < V THEN ELSE
               BEGIN
                 V := ABS(WCHARRY[LL+VALIA]);
                 L := WCHIA[I];
               END;
           END;
         MC := * + 1;
```

```
        WCHARRY[LL+L] := 0.0;
        FOR M := 1 STEP 1 UNTIL WCHLS DO
          BEGIN
            J := WCHIA[M];
            IF J = L THEN GO TO ENDM;
            IF ABS(WCHARRY[(L-1)*NRVARS+J]) < 0.80 THEN ELSE
              BEGIN
                MC := * + 1;
                WRITE(LINE,<"VAR "I2," REMOVED DUE TO HI CORRELATION",
                      " WITH VAR ",I2>,J,L);
                FOR N := 1 STEP 1 UNTIL NRVARS DO
                    IF N = J THEN ELSE
                        BEGIN
                          R[J] := 0.0017;
                          WCHARRY[(N-1)*NRVARS+J] := 0.0017;
                          WCHARRY[(J-1)*NRVARS+N] := 0.0017;
                        END;
              END;
ENDM:     END;
       END;
OUTHI:
    FOR I := 1 STEP 1 UNTIL NRVARS DO
       WCHARRY[LL+I] := R[I];
  END HICOR;
 $ PAGE
PROCEDURE CORPRINT(WCHARRY);
ARRAY WCHARRY[*];
BEGIN
  INTEGER J,L,LL,LINCNT;
  LINCNT := 0;
  FOR L := 1 STEP 12 UNTIL 85 DO
    BEGIN
      WRITE(LINE,<X9,12(X4,"VAR"I2,X1)//>,FOR LL := L STEP 1 UNTIL L+11
            DO LL);
      FOR J := 1 STEP 1 UNTIL NRVARS DO
         BEGIN
           WRITE(LINE,<X2,"VAR",I2,X1,12(F10.5)>,J,FOR LL := L STEP 1
                UNTIL L+11 DO WCHARRY[(J-1)*NRVARS+LL]);
           LINCNT := * + 1;
           IF LINCNT > 50 THEN
             BEGIN
               WRITE(LINE[SKIP 1]);
               LINCNT := 0;
               WRITE(LINE,<X9,12(X4,"VAR"I2,X1)//>,FOR LL := L STEP 1
                  UNTIL L+11 DO LL);
             END;
         END;
       WRITE(LINE[SPACE 2]);
       LINCNT := * + 3;
     END;
  WRITE(LINE[SKIP 1]);
END CORPRINT;
%****************START PROGRAM**********************
      NRVARS := 96;
      BINARY.OPEN := TRUE;
      B := BINARY.MAXRECSIZE;
      READ(CRD,CARDIN,BOARD,GRBD,CUTSCORE[3],CUTSCORE[2],CUTSCORE[1]);
      VAR := IF BOARD = "R" THEN 1 ELSE          %REGULAR
             IF BOARD = "N" THEN 2 ELSE          %NOMINATION
             IF BOARD = "P" THEN 3 ELSE          %PERMANENT
```

```
        4;                                            %TEMPORARY
    READ(BOARDFILE,640,BF);
    BDCNT := * + 1;
READBIN:
    READ(BINARY,B,BIN)[ENDRD];
    BINCNT := * + 1;
%   IF BINCNT > 1200 THEN GO TO ENDRD;
COMPARE:
    IF BF[1] > BIN[0] FOR 9 THEN GO TO READBIN;
    IF BF[1] < BIN[0] FOR 9 THEN GO TO READBD;
    IF BF[42] = "R" THEN GO TO READBIN;               %REPAT
    IF BF[581] > "99" OR BF[581] < "01" THEN GO TO READBIN;
    IF BF[593] = "000" THEN GO TO READBIN;       %BD SCORE
    ZONECNT[ZONEDEF] := * + 1;
    GO TO PASS1;
READBD:
    READ(BOARDFILE,640,BF)[ENDRD:PARR];
    BDCNT := * + 1;
    GO TO COMPARE;
PASS1:
    SCORE := INTEGER(BF[37],3) - 300;              %PRED SCORE
    BDSCR := INTEGER(BF[593],3);                   %BOARD SCORE
    IF BF[37] > "500" OR BF[37] < "300"            %PRED SCORE
        THEN GO TO CKSCORE;
    CUT := IF BF[37] > POINTER(CUTSCORE[ZONEDEF]) + 3 FOR 3 THEN 0
       ELSE IF BF[37] = POINTER(CUTSCORE[ZONEDEF]) + 3 FOR 3 THEN 1
       ELSE 2;      %0=PRED>CUT,1=PRED=CUT,2=PRED<CUT
    STATDIST1[ZONEDEF,SCORE,0] := * + 1;
    CASE VAR OF
        BEGIN
          1:  IF BF[249] = "N" THEN                     %REG APPT
                BEGIN                                     %REG SELECT
                  REPLACE BINSEL BY "X";
                  IX1 := 0;
                  STATDIST1[ZONEDEF,SCORE,1] := * + 1;
                END ELSE
                  IX1 := 1;                               %NON SELECT
          2:  IF BF[158] = "3" THEN                     %NOMINATION
                BEGIN                                     %NOM SELECT
                  REPLACE BINSEL BY "X";
                  IX1 := 0;
                  STATDIST1[ZONEDEF,SCORE,1] := * + 1;
                END ELSE
                  IX1 := 1;                               %NON SELECT
          3:  IF BF[156] = "X" THEN                     %PERMANENT
                BEGIN                                     %PERM SELECT
                  REPLACE BINSEL BY "X";
                  IX1 := 0;
                  STATDIST1[ZONEDEF,SCORE,1] := * + 1;
                END ELSE
                  IX1 := 1;                               %NON SELECT
          4:  IF BF[157] = "X" THEN                     %TEMPORARY
                BEGIN                                     %TEMP SELECT
                  REPLACE BINSEL BY "X";
                  IX1 := 0;
                  STATDIST1[ZONEDEF,SCORE,1] := * + 1;
                END ELSE
                  IX1 := 1;                               %NON SELECT
        END;
    STATDIST2[ZONEDEF,IX1,CUT] := * + 1;
```

```
CKSCORE:
      REPLACE BINPAN BY BF[581] FOR 2;                          %BD PANEL
      REPLACE BIN[124] BY BF[593] FOR 3;                        %BD SCORE
      BDSCRSQ := BDSCR * BDSCR;
      DIST[ZONEDEF,INTEGER(BF[581],2),0] := * + BDSCR;    %BD SCORE
      DIST[ZONEDEF,100,0] := * + BDSCR;
      DIST[ZONEDEF,INTEGER(BF[581],2),1] := * + BDSCRSQ;
      DIST[ZONEDEF,100,1] := * + BDSCRSQ;
      DIST[ZONEDEF,INTEGER(BF[581],2),2] := * + 1;
      DIST[ZONEDEF,100,2] := * + 1;
      WRITE(BINARY,B,BIN);
      GO TO READBIN;
PARR:
   PARCNT := * + 1; BOO := READ(BOARDFILE,640,BF); GO TO READBIN;
ENDRD:
      REWIND(BINARY);
      WRITE(LINE[SPACE 2],<"PREPARED(MMDDYY): ",A6>,TIME(15));
      WRITE(LINE[SPACE 2],<"BOARD: ",A1,A3>,BOARD,GRBD);
      FOR X := 1 STEP 1 UNTIL 3 DO
         BEGIN
            IF ZONECNT[X] = 0 THEN GO TO ENDSTAT;
            WRITE(LINE,<A6,A6>,TITL[X*12],TITL[X*12+6]);
            WRITE(LINE[SPACE 2],HDR3);
            HOLD1 := HOLD2 := 0;
            FOR IX1 := 0,1 DO
                 FOR CUT := 0 STEP 1 UNTIL 2 DO
                     HOLD2 := * + STATDIST2[X,IX1,CUT];
            HOLD1 := STATDIST2[X,0,0] + STATDIST2[X,1,2] +
                     STATDIST2[X,0,1];
            WRITE(LINE[SPACE 3],DET1,STATDIST2[X,0,0],
                  STATDIST2[X,0,1],STATDIST2[X,0,2],
                  STATDIST2[X,1,0],STATDIST2[X,1,1],
                  STATDIST2[X,1,2],HOLD1,HOLD1/HOLD2);
ENDSTAT: END;
      WRITE(LINE[SKIP 1]);
      CASE VAR OF
           BEGIN
             1:  WRITE(LINE,HDR1R);                           %REG APPT
                 WRITE(LINE[SPACE 2],HDR2R);
                 FOR SCORE := 0 STEP 1 UNTIL 200 DO
                      WRITE(LINE,<X4,I3,X9,3(X4,I4,X7,I4,X7)>,SCORE+300,
                          STATDIST1[1,SCORE,0],STATDIST1[1,SCORE,1],
                          STATDIST1[2,SCORE,0],STATDIST1[2,SCORE,1],
                          STATDIST1[3,SCORE,0],STATDIST1[3,SCORE,1]);
             2:  WRITE(LINE,HDR1N);                           %NOMINATION
                 WRITE(LINE[SPACE 2],HDR2N);
                 FOR SCORE := 0 STEP 1 UNTIL 200 DO
                      WRITE(LINE,<X4,I3,X13,I4,X7,I4>,SCORE+300,
                          STATDIST1[1,SCORE,0],STATDIST1[1,SCORE,1]);
             3:  WRITE(LINE,HDR1N);                           %PERMANENT
                 WRITE(LINE[SPACE 2],HDR2N);
                 FOR SCORE := 0 STEP 1 UNTIL 200 DO
                      WRITE(LINE,<X4,I3,X13,I4,X7,I4>,SCORE+300,
                          STATDIST1[1,SCORE,0],STATDIST1[1,SCORE,1]);
             4:  WRITE(LINE,HDR1T);                           %TEMPORARY
                 WRITE(LINE[SPACE 2],HDR2T);
                 FOR SCORE := 0 STEP 1 UNTIL 200 DO
                      WRITE(LINE,<X4,I3,X9,2(X4,I4,X7,I4,X7)>,SCORE+300,
                          STATDIST1[1,SCORE,0],STATDIST1[1,SCORE,1],
                          STATDIST1[2,SCORE,0],STATDIST1[2,SCORE,1]);
```

```
        END;
    FOR ZONE := 1 STEP 1 UNTIL 3 DO
        BEGIN
          IF ZONECNT[ZONE] = 0 THEN GO TO ENDLOOP;
          FOR PANEL := 1 STEP 1 UNTIL 100 DO
              BEGIN
                SAMPL := DIST[ZONE,PANEL,2];
                SUMM := DIST[ZONE,PANEL,0];
                IF SAMPL > 0 THEN
                    BEGIN
                      HOLD1 := (SAMPL * DIST[ZONE,PANEL,1]) -
                           (SUMM * SUMM);
                      HOLD2 := HOLD1/(SAMPL * SAMPL);
                      DIST[ZONE,PANEL,1] := SQRT(HOLD2);
                      DIST[ZONE,PANEL,0] := SUMM/SAMPL;
                    END;
              END;
ENDLOOP:    END;
READAGAIN:
    READ(BINARY,B,BIN)[ENDAGAIN];
    BINCNT1 := * + 1;
%   IF BINCNT1 > 1200 THEN GO TO ENDAGAIN;
    IF BINPAN > "99" OR BINPAN < "01" THEN GO TO READAGAIN;
    IF BIN[124] = "000" THEN GO TO READAGAIN;  %BOARD SCORE
    REPLACE BIN[127] BY                          %ADJ BOARD SCORE VAR87
      INTEGER(BIN[124],3) - ((DIST[ZONEDEF,INTEGER(BINPAN,2),0]
      - DIST[ZONEDEF,100,0])) FOR 3 DIGITS;
    WRITE(BINARY,B,BIN);
%**********THIS IS THE NEW
    ZNCNT[ZONEDEF] := * + 1;
    FOR X := 1 STEP 1 UNTIL 100 DO RECHOLD[X] := 0;
    X := 1;
    FOR I := 32 STEP 1 UNTIL 40,41,43,45 STEP 1 UNTIL 98,99 STEP 2
      UNTIL 107,109 STEP 1 UNTIL 123,124,127,130 STEP 1 UNTIL 134,
      135,137,139,140 DO
        BEGIN
          VARAMT := IF I=41 OR I=43 THEN INTEGER(BIN[I],2) ELSE
                    IF I >98 AND I <108 THEN INTEGER(BIN[I],2)/10 ELSE
                    IF I=124 OR I=127 THEN INTEGER(BIN[I],3) ELSE
                    IF I=135 OR I=137 THEN INTEGER(BIN[I],2)/10 ELSE
                      INTEGER(BIN[I],1);
          RECHOLD[X] := VARAMT;
          IF VARAMT > 0 THEN
            BEGIN
              IF ZONEDEF = 1 THEN                        %PRIMARY ZONE
Accumulate    BEGIN
data for each   SUMP[X] := * + VARAMT;
observation     SUMPS[X] := * + VARAMT**2;
              END ELSE
              IF ZONEDEF = 2 THEN                        %SECONDARY
                BEGIN
                  SUMS[X] := * + VARAMT;
                  SUMSS[X] := * + VARAMT**2;
                END ELSE
              IF ZONEDEF = 3 THEN                        %TERTIARY
                BEGIN
                  SUMT[X] := * + VARAMT;
                  SUMTT[X] := * + VARAMT**2;
                END;
            END;
```

```
              X := * + 1;
            END;
          FOR X := 1 STEP 1 UNTIL NRVARS-1 DO
            BEGIN
              VARAMT := RECHOLD[X+1];
              IF VARAMT = 0 THEN ELSE
                BEGIN
                  M := X*NRVARS;
                  FOR K := 1 STEP 1 UNTIL X DO
                    IF RECHOLD[K] > 0 THEN
                       BEGIN
                         IF ZONEDEF = 1 THEN
                            XYSUMP[M+K] := * + (VARAMT*RECHOLD[K]) ELSE
                         IF ZONEDEF = 2 THEN
                             XYSUMS[M+K] := * + (VARAMT*RECHOLD[K]) ELSE
                          IF ZONEDEF = 3 THEN
                             XYSUMT[M+K] := * + (VARAMT*RECHOLD[K]);
                       END;
                END;
            END;
        GO TO READAGAIN;
  ENDAGAIN:
        FOR X := 1 STEP 1 UNTIL 3 DO
            BEGIN
               IF ZONECNT[X] = 0  THEN GO TO ENDLOOP2;
               WRITE(LINE[SKIP 1]);
               WRITE(LINE[SPACE 2],HDR4,TITL[X*12],TITL[X*12+6]);
               FOR PANEL := 1 STEP 1 UNTIL 100 DO
                 IF DIST[X,PANEL,2] > 0 THEN
                  WRITE(LINE,DET4,PANEL,DIST[X,PANEL,0],
                        DIST[X,PANEL,1],DIST[X,PANEL,2]);
  ENDLOOP2: END;
  %********NEW
          XX := NRVARS DIV 8; XY := NRVARS MOD 8;
          IF ZNCNT[1] > 0 THEN
            BEGIN
              M1 := ZNCNT[1] -1;
              FOR X := 1 STEP 1 UNTIL NRVARS DO
```

compute mean and standard deviation

```
                BEGIN
                  MEANP[X] := SUMP[X]/ZNCNT[1];
                  STDVP[X]:=(SUMPS[X]-ZNCNT[1]*(MEANP[X]**2))/M1;
                  IF STDVP[X] > 0 THEN STDVP[X] := STDVP[X]**0.5;
                END;
              FOR X := 0 STEP 1 UNTIL XX -1 DO
                 WRITE(PRI,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL 8 DO
                    MEANP[X*8+K]);
              IF XY > 0 THEN
                  WRITE(PRI,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL XY DO
                     MEANP[XX*8+K]);
              FOR X := 0 STEP 1 UNTIL XX -1 DO
                 WRITE(PRI,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL 8 DO
                     STDVP[X*8+K]);
              IF XY > 0 THEN
                 WRITE(PRI,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL XY DO
                    STDVP[XX*8+K]);
            END;
          IF ZNCNT[2] > 0 THEN
            BEGIN
              M2 := ZNCNT[2] -1;
              FOR X := 1 STEP 1 UNTIL NRVARS DO
```

```
          BEGIN
            MEANS[X] := SUMS[X]/ZNCNT[2];
            STDVS[X]:=(SUMSS[X]-ZNCNT[2]*(MEANS[X]**2))/M2;
            IF STDVS[X] > 0 THEN STDVS[X] := STDVS[X]**0.5;
          END;
%**WRITE MEANS
        FOR X := 0 STEP 1 UNTIL XX -1 DO
          WRITE(SEC,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL 8 DO
                MEANS[X*8+K]);
        IF XY > 0 THEN
          WRITE(SEC,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL XY DO
                MEANS[XX*8+K]);
        FOR X := 0 STEP 1 UNTIL XX -1 DO
          WRITE(SEC,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL 8 DO
                STDVS[X*8+K]);
        IF XY > 0 THEN
          WRITE(SEC,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL XY DO
                STDVS[XX*8+K]);
      END;
    IF ZNCNT[3] > 0 THEN                              %TERTIARY
      BEGIN
        M3 := ZNCNT[3] - 1;
        FOR X := 1 STEP 1 UNTIL NRVARS DO
          BEGIN
            MEANT[X] := SUMT[X]/ZNCNT[3];
            STDVT[X]:=(SUMTT[X]-ZNCNT[3]*(MEANT[X]**2))/M3;
            IF STDVT[X] > 0 THEN STDVT[X] := STDVT[X]**0.5;
          END;
        FOR X := 0 STEP 1 UNTIL XX -1 DO
          WRITE(TER,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL 8 DO
                MEANT[X*8+K]);
        IF XY > 0 THEN
          WRITE(TER,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL XY DO
                MEANT[XX*8+K]);
        FOR X := 0 STEP 1 UNTIL XX -1 DO
          WRITE(TER,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL 8 DO
                STDVT[X*8+K]);
        IF XY > 0 THEN
          WRITE(TER,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL XY DO
                STDVT[XX*8+K]);
      END;
    FOR I := 2 STEP 1 UNTIL NRVARS DO
      BEGIN
        L := (I-1) * NRVARS;
        TEMP := SUMP[I]; TEMS := SUMS[I]; TEMT := SUMT[I];
        IF ZNCNT[1] > 0 THEN                 %PRIMARY ZONE
          TVAP := STDVP[I];
        IF ZNCNT[2] > 0 THEN                 %SEC ZONE
          TVAS := STDVS[I];
        IF ZNCNT[3] > 0 THEN                 %TER ZONE
          TVAT := STDVT[I];
        FOR J := 1 STEP 1 UNTIL I-1 DO
          BEGIN
            M := L+J;
            IF ZNCNT[1] > 0 THEN             %PRIMARY ZONE
              BEGIN
                VAP := TVAP*STDVP[J];
                IF VAP > 0 THEN
                  BEGIN
                    XYSUMP[M] := ((XYSUMP[M]-(TEMP*SUMP[J]/ZNCNT[1]))/
```

```
          M1)/VAP;
        IF ABS(XYSUMP[M]) < 0.80 THEN ELSE
        IF I = 86 OR I = 87 THEN ELSE       %DEP VARIABLE
        IF J = 86 OR J = 87 THEN ELSE
          BEGIN
            ISW := 0;
            FOR MRPRI := 1 STEP 1 UNTIL LSPRI+1 DO
               IF IAPRI[MRPRI] = I THEN ISW := 1;
            IF ISW = 1 THEN ELSE
               BEGIN
                 LSPRI := * + 1;
                 IAPRI[LSPRI] := I;
               END;
            ISW := 0;
            FOR MRPRI := 1 STEP 1 UNTIL LSPRI+1 DO
               IF IAPRI[MRPRI] = J THEN ISW := 1;
            IF ISW = 1 THEN ELSE
               BEGIN
                 LSPRI := * + 1;
                 IAPRI[LSPRI] := J;
               END;
          END;
       END ELSE XYSUMP[M] := 0.0019;
     XYSUMP[((J-1)*NRVARS)+I] := XYSUMP[M];
     END;
  IF ZNCNT[2] > 0 THEN          %SEC ZONE
     BEGIN
      VAS := TVAS*STDVS[J];
      IF VAS > 0 THEN
        BEGIN
        XYSUMS[M] := ((XYSUMS[M]-(TEMS*SUMS[J]/ZNCNT[2]))/
          M2)/VAS;
        IF ABS(XYSUMS[M]) < 0.80 THEN ELSE
        IF I = 86 OR I = 87 THEN ELSE       %DEP VARIABLE
        IF J = 86 OR J = 87 THEN ELSE
          BEGIN
            ISW := 0;
            FOR MRSEC := 1 STEP 1 UNTIL LSSEC+1 DO
               IF IASEC[MRSEC] = I THEN ISW := 1;
            IF ISW = 1 THEN ELSE
               BEGIN
                 LSSEC := * + 1;
                 IASEC[LSSEC] := I;
               END;
            ISW := 0;
            FOR MRSEC := 1 STEP 1 UNTIL LSSEC+1 DO
               IF IASEC[MRSEC] = J THEN ISW := 1;
            IF ISW = 1 THEN ELSE
               BEGIN
                 LSSEC := * + 1;
                 IASEC[LSSEC] := J;
               END;
          END;
       END ELSE XYSUMS[M] := 0.0019;
    XYSUMS[((J-1)*NRVARS)+I] := XYSUMS[M];
     END;
  IF ZNCNT[3] > 0 THEN          %TER ZONE
     BEGIN
      VAT := TVAT*STDVT[J];
      IF VAT > 0 THEN
```

compute the correlation matrix

```
            BEGIN
            XYSUMT[M] := ((XYSUMT[M]-(TEMT*SUMT[J]/ZNCNT[3]))/
               M3)/VAT;
            IF ABS(XYSUMT[M]) < 0.80 THEN ELSE
            IF I = 86 OR I = 87 THEN ELSE        %DEP VARIABLE
            IF J = 86 OR J = 87 THEN ELSE
              BEGIN
                ISW := 0;
                FOR MRTER := 1 STEP 1 UNTIL LSTER+1 DO
                   IF IATER[MRTER] = I THEN ISW := 1;
                IF ISW = 1 THEN ELSE
                   BEGIN
                     LSTER := * + 1;
                     IATER[LSTER] := I;
                   END;
                ISW := 0;
                FOR MRTER := 1 STEP 1 UNTIL LSTER+1 DO
                   IF IATER[MRTER] = J THEN ISW := 1;
                IF ISW = 1 THEN ELSE
                   BEGIN
                     LSTER := * + 1;
                     IATER[LSTER] := J;
                   END;
              END;
            END ELSE XYSUMT[M] := 0.0019;
          XYSUMT[((J-1)*NRVARS)+I] := XYSUMT[M];
         END;
      END;
      XYSUMP[L+I] := XYSUMS[L+I] := XYSUMT[L+I] := 1.0;
    END;
  XYSUMP[1] := XYSUMS[1] := XYSUMT[1] := 1.0;
  I := NRVARS**2;
  XYSUMP[I] := XYSUMS[I] := XYSUMT[I] := 1.0;
  IF ZNCNT[1] > 0 THEN
     BEGIN
       WRITE(LINE[SKIP 1]);
       WRITE(LINE,<"PRIMARY/PILOT ZONE CORRELATION MATRIX"/>);
       CORPRINT(XYSUMP);
       WRITE(LINE,<"PRIMARY/PILOT ZONE HICOR:"/>);
       HICOR(XYSUMP,IAPRI,LSPRI);
     END;
  IF ZNCNT[2] > 0 THEN
     BEGIN
       WRITE(LINE[SKIP 1]);
       WRITE(LINE,<"SECONDARY/NAV ZONE CORRELATION MATRIX"/>);
       CORPRINT(XYSUMS);
       WRITE(LINE,<"SECONDARY/NAV ZONE HICOR:"/>);
       HICOR(XYSUMS,IASEC,LSSEC);
     END;
  IF ZNCNT[3] > 0 THEN
     BEGIN
       WRITE(LINE[SKIP 1]);
       WRITE(LINE,<"TERTIARY/SPT ZONE CORRELATION MATRIX"/>);
       CORPRINT(XYSUMT);
       WRITE(LINE,<"TERTIARY/SPT ZONE HICOR:"/>);
       HICOR(XYSUMT,IATER,LSTER);
     END;
%**  XX = NRVARS DIV 8, XY= NRVARS MOD 8 (CONTROL NR LINES WRITTEN)
  FOR I := 0 STEP 1 UNTIL NRVARS-1 DO
    BEGIN
```

```
L := I * NRVARS;
IF ZNCNT[1] > 0 THEN
   BEGIN
     FOR X := 0 STEP 1 UNTIL XX-1 DO
        WRITE(PRI,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL 8 DO
            XYSUMP[L+(X*8+K)]);
     IF XY > 0 THEN
        WRITE(PRI,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL XY DO
            XYSUMP[L+(XX*8+K)]);
    END;
IF ZNCNT[2] > 0 THEN
   BEGIN
     FOR X := 0 STEP 1 UNTIL XX-1 DO
        WRITE(SEC,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL 8 DO
            XYSUMS[L+(X*8+K)]);
     IF XY > 0 THEN
        WRITE(SEC,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL XY DO
            XYSUMS[L+(XX*8+K)]);
   END;
IF ZNCNT[3] > 0 THEN
   BEGIN
     FOR X := 0 STEP 1 UNTIL XX-1 DO
        WRITE(TER,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL 8 DO
            XYSUMT[L+(X*8+K)]);
     IF XY > 0 THEN
        WRITE(TER,<8F10.5,X2>,FOR K := 1 STEP 1 UNTIL XY DO
            XYSUMT[L+(XX*8+K)]);
   END;
 END;
IF ZNCNT[1] > 0 THEN
   LOCK(PRI);
IF ZNCNT[2] > 0 THEN
   LOCK(SEC);
IF ZNCNT[3] > 0 THEN
   LOCK(TER);
WRITE(LINE[SKIP 1]);
WRITE(LINE[SKIP 1],<X61,"END REPORT">);
WRITE(LINE,<"PREPARED(MMDDYY): ",A6>,TIME(15));
WRITE(LINE,<" ERGE ",A1,A3>,BOARD,GRBD);
WRITE(LINE,<X10,"BOARD COUNT - ",I6>,BDCNT);
WRITE(LINE,<X10,"BIN REC IN - ",I6>,BINCNT);
WRITE(LINE,<X10,"BIN REC IN/OUT - ",I6>,BINCNT1);
WRITE(LINE,<X10,"RECORD COUNTS INPUT TO REG BUILD - ",
   3(I5,X2)>,FOR I := 1 STEP 1 UNTIL 3 DO ZNCNT[I]);
WRITE(LINE,<X10,"PROCESS TIME - ",F6.2," SECS">,TIME(2)/60);
WRITE(LINE,<X10,"I/O TIME - ",F6.2," SECS">,TIME(3)/60);
WRITE(LINE,<X10,"PARITY ERR CNT:",I3>,PARCNT);
END.
```